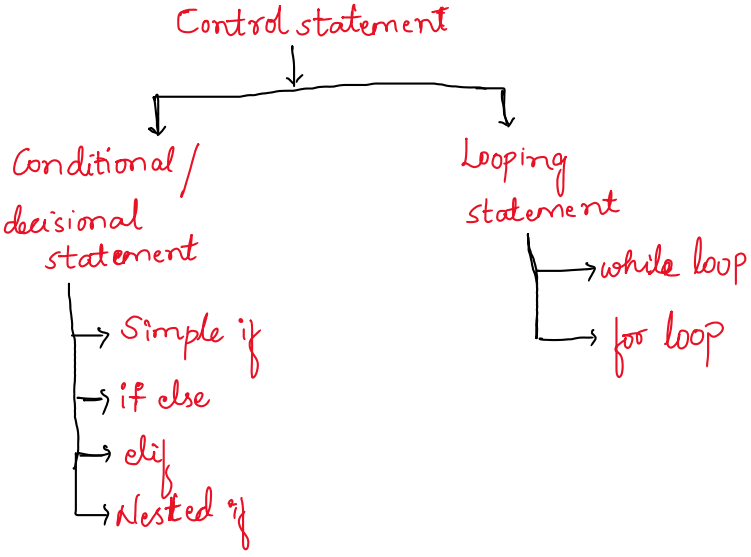
06 February 2025 08:13

Day-20

Control Statement:
--- It is used to control the flow of execution.

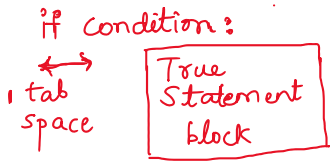
Types:



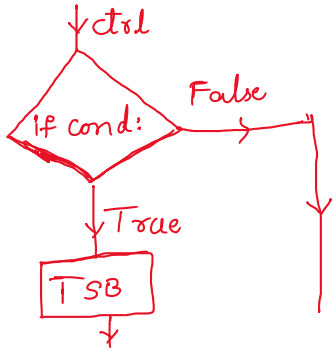
Conditional statement:
--- It is used to control the flow of execution based on conditions.

1) **Simple if:**
--- It is a keyword which is used to check the condition and it will execute the statement block if the condition is True. It will ignore the statement block if the condition is False.

Syntax:



Flow diagram:



Programs:

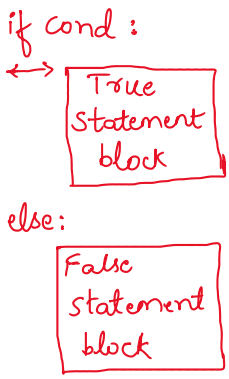
```
#Simple if

#WAP to check whether the number is even.
'''
num=int(input('Enter the number: '))
if num%2==0:
   print(num,'is even')'''

#WAP to check whether string has exactly 5 characters in it .
'''
s=input('Enter the string: ')
if len(s)==5:
   print(s,'has exactly 5 characters in it')'''
```

```
#WAP to check whether the number is greater than 200 .
'''
num=int(input('Enter the number: '))
if num>200:
   print(num,'is greater than 200')'''

#WAP to print the square of the number if the number is multiple of 3.
'''
num=int(input('Enter the number: '))
if num%3==0:
   print(num**2)'''

#WAP to check whether the character is Uppercase .
'''
s=input('Enter the character: ')
if 'A'<=s<='Z':
   print(s,'is uppercase')'''
```

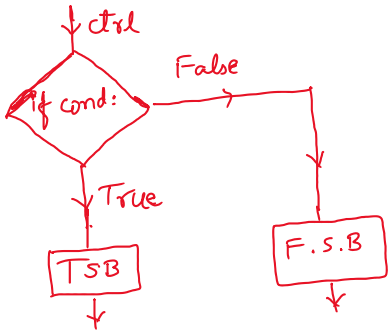
2) if else:

--- It is an advance version of simple if. It will execute the True statement block if the condition is True. It will execute the False statement block if the condition is False.

Syntax:



Flow diagram:



Programs:

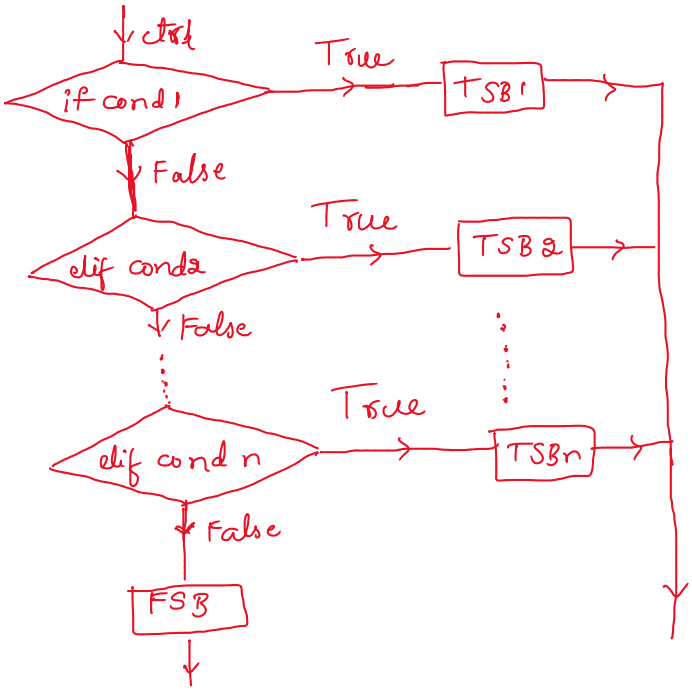
```
#WAP to check the given data is float or not.
'''
data=eval(input('Enter the data: '))
if type(data)==float:
   print('given data is float')
else:
   print('given data is not float')'''

#WAP to check whether the string is palindrome or not
'''
s=input('Enter the string: ')
if s==s[::-1]:
   print(s,'is palindrome')
else:
   print(s,'is not palindrome')'''

#WAP to check whether the given character is vowel or not.
'''
s=input('Enter the character: ')
if s in 'aeiouAEIOU':
   print(s,'is vowel')
else:
   print(s,'is not vowel')'''
```

```
#WAP to check whether the given data is single valued datatype or not.
'''
data=eval(input('Enter the data: '))
if type(data) in [int, float, complex, bool] :
   print(data,'is SVDT')
else:
   print(data,'is not SVDT')'''

'''
data=eval(input('Enter the data: '))
if type(data) == int or type(data) == float or  type(data) == complex or type(data) == bool:
   print(data,'is SVDT')
else:
   print(data,'is not SVDT')'''


#WAP to check whether the given integer is 3 digit number or not.
'''
num = abs(int(input('Enter the number: ')))
if 100<=num<=999:
   print(num, 'is 3 digit number')
else:
   print(num,'is not 3 digit number')'''
```

3) elif : Whenever we want to check multiple condition and execute multiple statement block of each and every condition then elif is used.

Syntax:

Flow Diagram:



Programs:

```
#elif

# WAP to find the relation between two integers.
'''
num1=int(input('Enter the number1: '))
num2=int(input('Enter the number2: '))
if num1>num2:
   print(num1,'is greater')
elif num1<num2:
```

```
    print(num1,'is lesser')
else:
    print(num1,'is equal to',num2)'''

# WAP to check whether the character is uppercase or lowercase or digit or special characters.
'''
ch=input('Enter the character: ')
if 'A'<=ch<='Z':
    print(ch,'is uppercase')
elif 'a'<=ch<='z':
    print(ch,'is lowercase')
elif '0'<=ch<='9':
    print(ch,'is digit')
else:
    print(ch,'is special character')'''

# WAP to check whether the given integer is single digit or two digit or three digit or more than 3 digit
'''
num=abs(int(input('Enter the number: ')))
if 0<=num<=9:
    print('single digit')
elif 10<=num<=99:
    print('two digit')
elif 100<=num<=999:
    print('three digit')
elif num>999:
    print('more than 3 digit')'''

# WAP to find the greatest among 4 numbers
'''
a,b,c,d=int(input('Enter the num1: ')),int(input('Enter the num2: ')),int(input('Enter the num3: ')),int(input('Enter the
num4: '))
if a>b and a>c and a>d:
    print(a,'is greatest')
elif b>a and b>c and b>d:
    print(b,'is greatest')
elif c>a and c>b and c>d:
    print(c,'is greatest')
else:
    print(d,'is greatest')'''
```

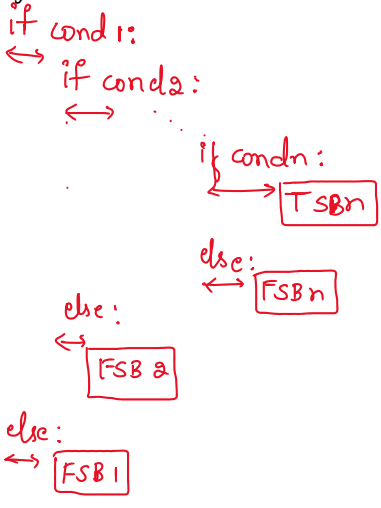
4) Nested if:

--- Whenever it is necessary to check a condition before another condition we use Nested if.

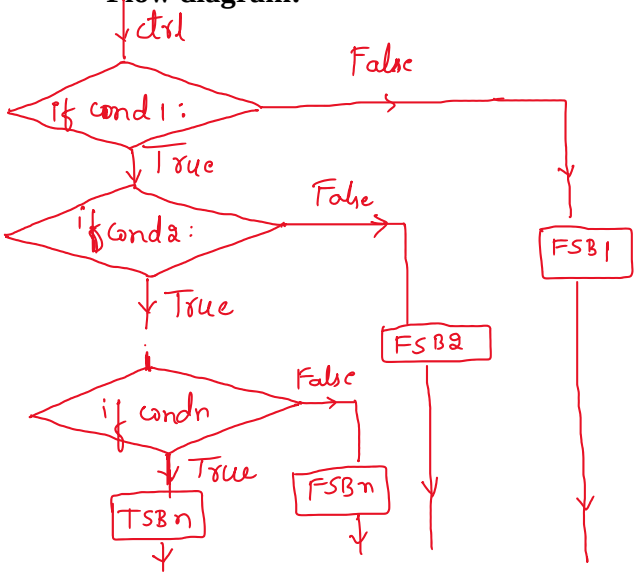
Or

Condition inside a condition is referred as Nested if.

Syntax:



Flow diagram:

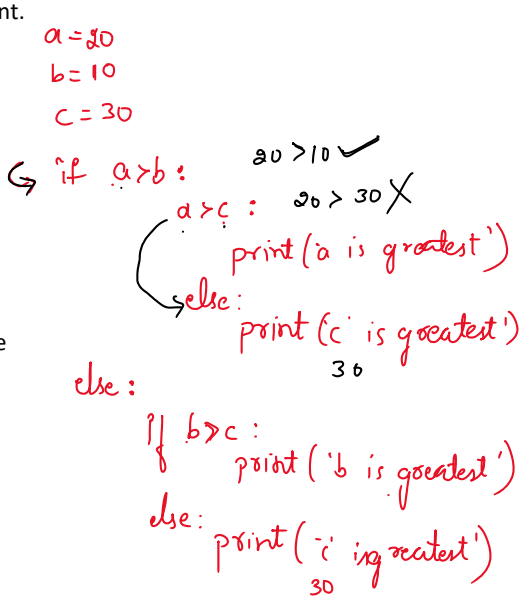


Programs:

#Nested if

#WAP to check whether the given character is vowel or consonant.

```
'''
ch=input('Enter the character: ')
if 'a'<=ch<='z' or 'A'<=ch<='Z':
   if ch in 'aeiouAEIOU':
      print(ch,'is vowel')
   else:
      print(ch,'is consonant')
else:
   print(ch,'is not alphabet')'''
```



#WAP to login to the instagram by entering the proper username and password.

```
'''
username='__seclusive__'
password='saku@123'
un=input('Enter your username: ')
if un == username:
   pw=input('Enter your password: ')
   if pw == password:
      print('Login Successful')
   else:
      print('Incorrect password')
else:
   print('Invalid username')'''
```

WAP to find the greatest among three numbers.

```
'''
a,b,c=int(input('Enter the num1: ')),int(input('Enter the num2: ')),int(input('Enter the num3: '))
if a>b:
   if a>c:
      print(a,'is greater')
   else:
      print(c,'is greater')
else:
   if b>c:
      print(b,'is greater')
   else:
      print(c,'is greater')'''
```

Assignment:

WAP to find the greatest among four numbers.

Day-21

Looping Statement:

--- It is also used to control the flow of execution by executing the same set of instructions again and again.

Types:

- **While loop**
- **For loop**

1) While loop:

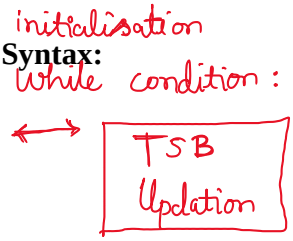
--- It is used to execute the same set of instructions again and again until the condition become

False.

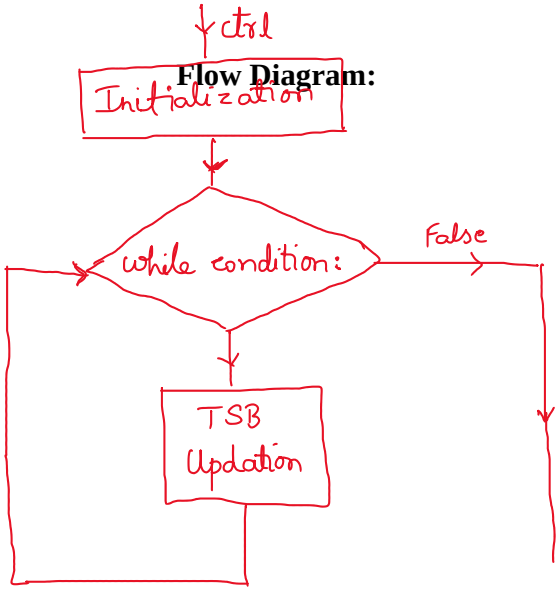
Note: There are 2 manditory / compulsory things in while loop

- Initialisation
- Updation

Syntax:

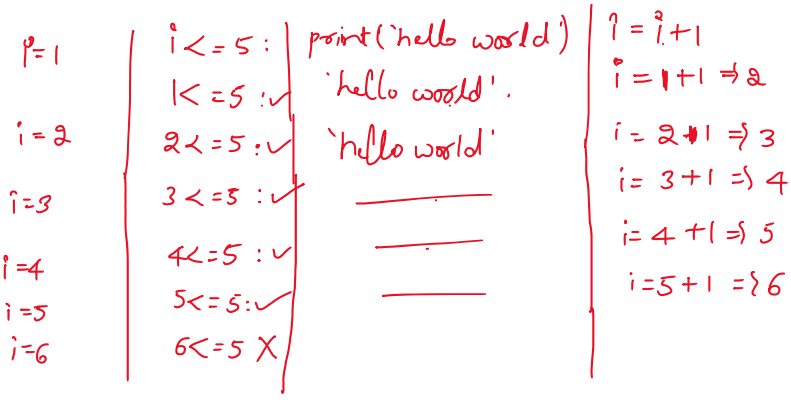


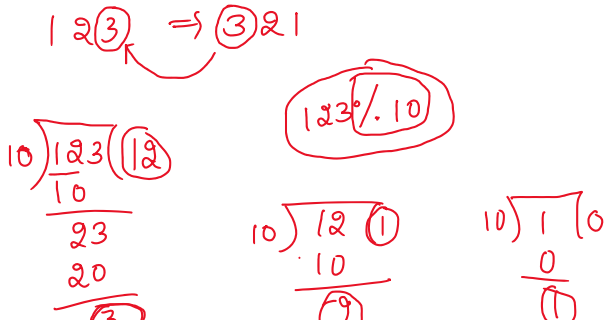
Flow Diagram:



Programs:

```
# While loop

#WAP to print hello world for 5 times.
'''
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')'''

''''
i=1
while i<=5:
   print('hello world')
   i=i+1'''


#WAP to print the n natural numbers
'''
num = int(input('Enter the number: '))
i=1
while i<=num:
   print(i)
   i+=1'''

# WAP to print the even numbers from 1 to 30
'''
num = int(input('Enter the number: '))
i=1
while i<=num:
   if i%2==0:
      print(i)
   i+=1'''
```



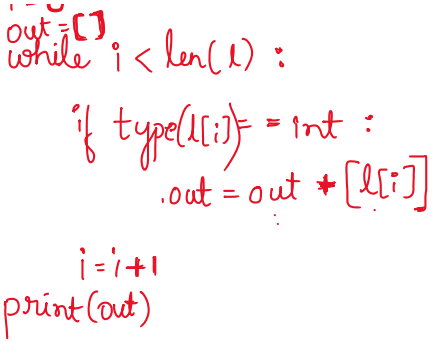


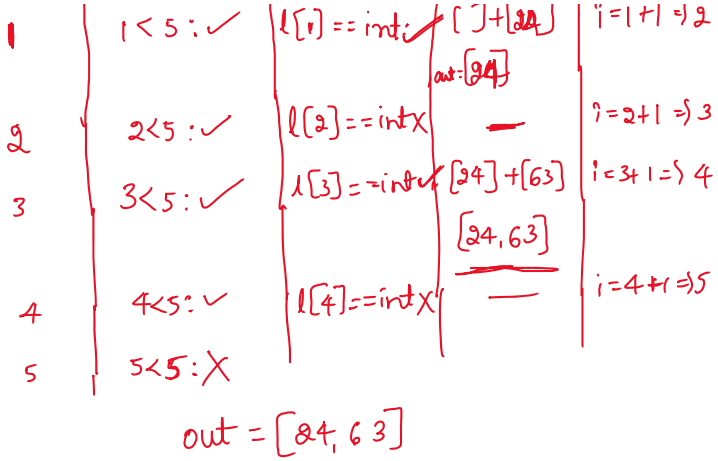
```
    if i%2==0:
        print(i)
    i+=1'''

'''
num = int(input('Enter the number: '))
i=2
while i<=num:
    print(i)
    i=i+2'''

# WAP to print the number from n to 1.
'''
num = int(input('Enter the number: '))
i=num
while i>=1:
    print(i)
    i=i-1'''


# WAP to reverse the given number without typecasting.
'''
n = int(input('Enter the number: '))
res=0
while n>0:
    rem = n%10
    res = res + rem
    n = n//10
print(res)'''

#WAP to print the sum of n natural numbers.
'''
n = int(input('Enter the number: '))
out = 0
i=0
while i<=n:
     out = out + i
     i = i + 1
print(out)'''

#WAP to print the product of n natural numbers.
'''
n = int(input('Enter the number: '))
out = 1
i=1
while i<=n:
     out = out * i
     i = i + 1
print(out)'''

# WAP to print the every single character in a string.
'''
s = input('Enter the string: ')
i=0
while i< len(s):
    print(s[i])
    i+=1'''
```

Day-23

Continuation of while loop programs,

```
# WAP to extract the integers from the list.
```





Practical proof:

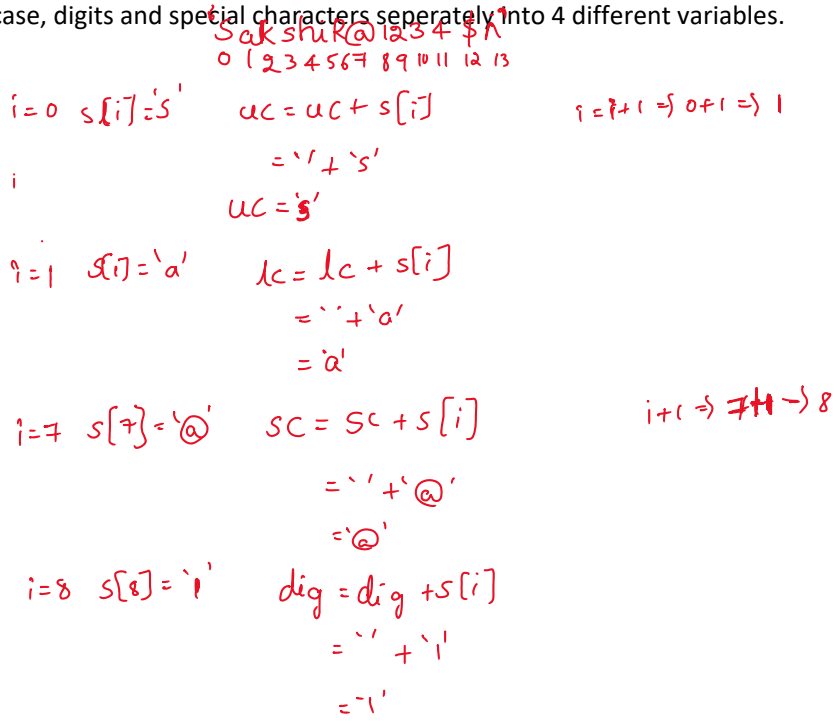
```
l = eval(input('Enter the list: '))
i = 0
out = []
while i < len(l):
   if type(l[i]) == int:
      out = out + [l[i]]
   i = i + 1
print(out)
```

Or

```
l = eval(input('Enter the list: '))
i = 0
out = []
while i < len(l):
   if type(l[i]) == int:
      out.append(l[i])
   i = i + 1
print(out)
```

WAP to extract uppercase, lowercase, digits and special characters seperately into 4 different variables.

```
s  = input('Enter the string: ')
uc = ''
lc = ''
dig = ''
sc = ''
i = 0
while i < len(s):
   if 'A'<=s[i]<='Z':
      uc = uc + s[i]
   elif 'a'<=s[i]<='z':
      lc = lc + s[i]
   elif '0'<=s[i]<='9':
      dig = dig + s[i]
   else:
      sc = sc + s[i]
   i = i + 1
print(uc)
print(lc)
print(dig)
print(sc)
```



Output:

```
Enter the string: SakshiR@1234$^
SR
akshi
1234
@$^
```

WAP to find the sum of integers in the tuple.

```
t = eval(input('Enter the tuple: '))
add = 0
i = 0
while i < len(t):
    if type(t[i]) == int:
        add = add + t[i]
    i = i + 1
print(add)
```

len(t) = 5

i	0< len(t)	type(t[i])==int :	add+=t[i]	i=i+1
0	0<5 : ✓	t[0]==int ✓	add+t[0]	i=0+1
			0+10	i=1
			add=10	
1	1<5: ✓	t[1]==int X	—	i=1+1
				i=2
2	2<5: ✓	t[2]==int: ✓	add+t[2]	i=2+1
			10+40	i=3
			add=50	
3	3<5 : ✓	t[3]==int X	—	i=3+1
				i=4
4	4<5 : ✓	t[4]==int X	—	i=4+1
				i=5
5	5<5 X			

add = 5

Output:

```
Enter the tuple: (10,2.3,40,True,'bye')
50
```

From Uppercase to lowercase conversion:

```
ord('A')
65
ord('a')
97
chr(97)
'a'
chr(ord('A')+32)
'a'
chr(ord('A')+32)
'a'
chr(ord('B')+32)
'b'
chr(ord('C')+32)
'c'
```

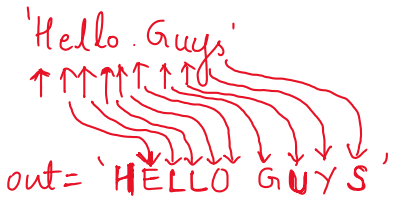
From Lowercase to Uppercase conversion:

```
ord('A')
65
ord('a')
97
chr(ord('a')-32)
'A'
chr(ord('b')-32)
'B'
chr(ord('c')-32)
'C'
```

WAP to convert all the lowercase alphabet from the string to uppercase alphabet.

```
s=input('Enter the string: ')
out = ''
i=0
```

```
while i<len(s):
   if 'a'<=s[i]<='z':
      out = out + chr(ord(s[i])-32)
   else:
      out = out + s[i]
   i = i + 1
print(out)
```



Output:

```
Enter the string: Hello Guys
HELLO GUYS
```

Toggle:
--- Converting from lowercase to uppercase and uppercase to lowercase and keeping the rest of the characters as it is in the string.

```
# WAP to toggle the string.

s=input('Enter the string: ')
toggle = ''
i=0
while i<len(s):
   if 'a'<=s[i]<='z':
      toggle = toggle + chr(ord(s[i])-32)
   elif 'A'<=s[i]<='Z':
      toggle = toggle + chr(ord(s[i])+32)
   else:
      toggle = toggle + s[i]
   i = i + 1
print(toggle)
```

Output:

```
Enter the string: Python Is The Champion 21*^$^#$
pYTHON iS tHE cHAMPION 21*^$^#$
```

```
# WAP to find the product of all the float numbers present at the odd index in a given list.

l = eval(input('Enter the list: '))
prod = 1
i = 0
while i<len(l):
   if type(l[i])==float and i%2!=0:
      prod = prod * l[i]
   i = i + 1
print(prod)
```

Output:

```
Enter the list: [2.3,4.6,7.1,6.2,8.5]
28.52
4.6 * 6.2
28.52

Enter the list: [True,4.6,7.1,6.2,2+3j,'gm',2.6]
```

Day-24

For loop:

--- It is self-iterative loop.

Drawback of while loop :

- It requires index position of the values in collection to traverse but set and dictionary won't support for indexing.
- Initialization , condition and updation are manditory in while loop.

Advantage of for loop over while loop:

- No need of Initialization and updation.
- It can be used of all the collection datatypes.

range(): It is used to create the sequence of integers between the given values.

Syntax:

range(SV,EV+1,updation)
range(SV,EV- 1,updation)

Range shortcuts:

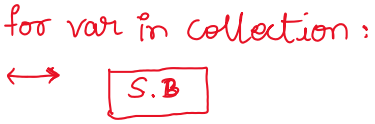
- If updation == +1
  range(SV, EV+- 1)

- If SV == 0 and updation == +1
  range(EV+1)

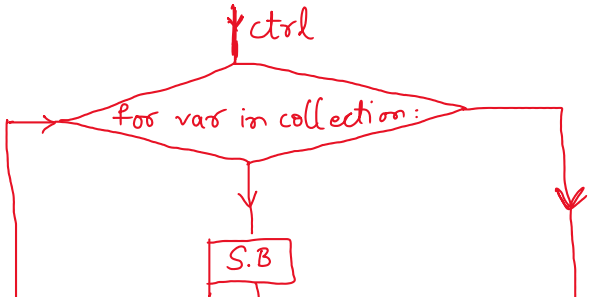
Key points -- To print the range of values use typecasting.

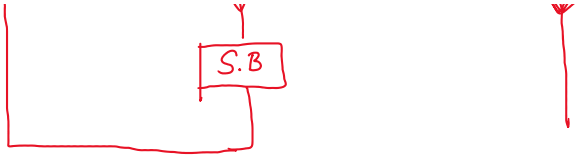
```
range(1,10+1)
range(1, 11)
list[range(1, 11)]
list[range(1, 11)]
list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Syntax:



Flow diagram:



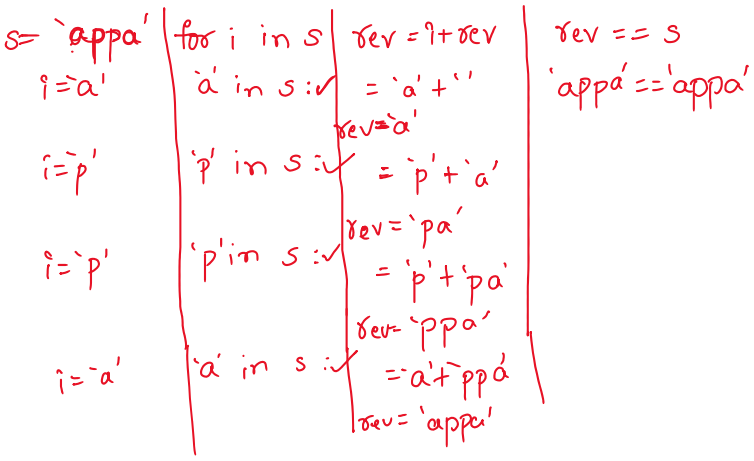


Programs:

```
'''
l = [10,2.3,6+7j,78]
for i in l:
   print(i)'''

'''
for i in {1,7,4.5,45,True}:
   print(i)'''
'''
for i in {'a':10,'b':20,'c':30}:
   print(i)'''

'''
for i in range(1,7,2):
   print(i)'''

# Actual programs on for loop

# To find the length of the given collection without using len() function.

c = eval(input('Enter the collection: '))
count = 0
for i in c:
   count += 1
print(count)


# WAP to extract vowels from the string.
'''
s = input('Enter the string: ')
out = ''
i = 0
while i<len(s):
   if s[i] in 'aeiouAEIOU':
      out+=s[i]
   i+=1
print(out)'''

'''
s = input('Enter the string: ')
out = ''
for i in s:
   if i in 'aeiouAEIOU':
      out+=i
print(out)'''

# WAP to replace space by an underscore in a given string.
'''
s = input('Enter the string: ')
rep = ''
for i in s:
   if i == ' ':
      rep += '_'
   else:
      rep += i
print(rep)'''
```

WAP to check whether the string is palindrome or not without using slicing.

```
s = input('Enter the string: ')
rev = ''
for i in s:
   rev = i + rev
print(rev)
if rev == s:
   print('palindrome')
else:
   print('not palindrome')
```



WAP to remove duplicates values from the list.

```
'''
l = eval(input('Enter the list: '))
out = []
for i in l:
   if i not in out:
      out.append(i)
print(out)'''
```

Get the following output

```
'''
 Input :     (12,3.4,'hello',8+9j,'python','bye',2.6,True)
 Output :   {'hello':5,'python':6,'bye':3}'''

'''
t = eval(input('Enter the tuple: '))
out = {}
for i in t:
   if type(i)==str:
      out[i]= len(i)
print(out)'''
```

Get the following output.

```
'''
Input : [12,'data',3.4,True,'science',3+7j,'engineer']
Output : {'data':'da','science':'se','engineer':'er'}'''

'''
l = eval(input('Enter the list: '))
out = {}
for i in l:
   if type(i) == str:
      out[i]=i[0]+i[-1]
print(out)'''
```

WAP to get the following output

```
'''
Input : 'ApPlE#23'
Output : {'A':a,'p':P,'P':p,'l':'L','E':e}'''

'''
s = input('Enter the string: ')
out = {}
for i in s:
   if 'A'<=i<='Z':
      out[i] = chr(ord(i)+32)
```

```
    elif 'a'<=i<='z':
        out[i] = chr(ord(i)-32)
print(out)'''

# WAP to get the following output
'''
Input : 'hai hello how are you'
Output : {'hai':3,'hello':5,'how':3,'are':3,'you':3}'''

'''
s = input('Enter the string: ')
out = {}
a = s.split()
for i in a:
    out[i] = len(i)
print(out)'''

# WAP to get the following output
'''
Input : 'hello fellow coders'
Output : 'olleh wollef sredoc'          '''

'''
s = input('Enter the string: ')
out = []
a = s.split()
for i in a:
    out.append(i[::-1])
print(''.join(out)) '''

# WAP to get the following output
'''
Input : 'abcabacbcc'
Output : 'a3b3c4'        '''

'''
s = input('Enter the string: ')
out = ''
for i in s:
    if i not in out:
        c = s.count(i)
        out += i + str(c)
print(out)'''
```

⇒ Tracing

s = 'abcabacbcc'

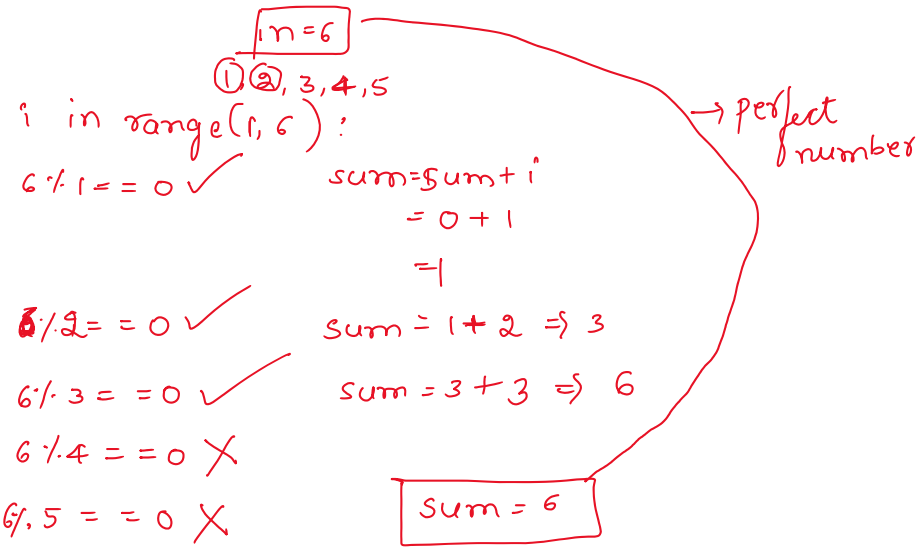
i in s	if i not in out:	c = s.count(i)	out += i + str(c)
i = a	a not in '' ✓	c = 3	= 'a' + '3'
			out = 'a3'
i = b	b not in 'a3' ✓	c = 3	= 'b' + '3'
			= 'a3' + 'b3'
			= 'a3b3'
i = c	c not in 'a3b3' ✓	c = 4	= 'a3b3' + 'c' + '4'
i = a	a not in 'a3b3c4' ✗		= 'a3b3c4'

out = 'a3b3c4'

What is perfect number?
--- If the sum of all the factors of a given number (excluding the number itself) becomes equal to the given number means , we can say the number is Perfect number.

```
# WAP to check whether the given number is perfect number or not.
'''
```

```
n = int(input('Enter the number: '))
sum = 0
for i in range(1,n):
   if n % i == 0:
      sum += i
if sum == n:
   print('perfect number')
else:
   print('not perfect number')'''
```



Day-25

Armstrong Number : We have to power each and every digit of the number by its length and add the value to a variable. If the added value becomes equal to the number itself means the number is armstrong number.

153 => 1**3 + 5**3 + 3**3 => 1+125+27 => 153

s='153'
len(s)=3

153 == 153

```
# WAP to find the given integer is armstrong number of not.
'''
n = int(input('Enter the number: '))
sum = 0
a = str(n)
for i in a:
   sum += int(i) ** len(a)
if sum == n:
   print('armstrong number')
else:
   print('not armstrong number')'''

# WAP to extract key-value pair from the dictionary only if key is of string type.
'''
d = eval(input('Enter the dictionary: '))
out = {}
for i in d:
   if type(i) == str:
      out[i] = d[i]
print(out)'''

# Get the following output
'''
Input : { 'M155': 'morning batch',   'E140':'Evening batch',    'M9':'weekend batch'}
output : { 'morning batch': 'M155',  'Evening batch':'E140',   'weekend batch':'M9'} '''

'''
d = eval(input('Enter the dictionary: '))
```

```
out = {}
for i in d:
   out[d[i]]=i
print(out)'''
```

Nested for loop:

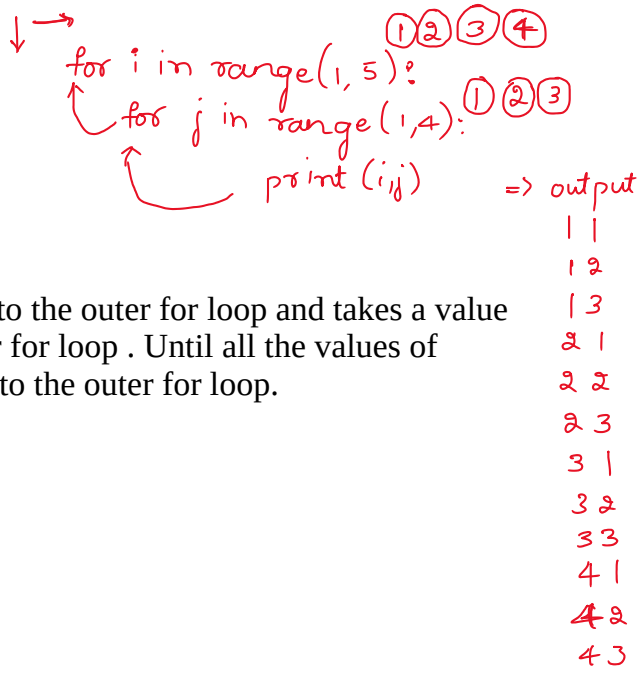
--- It is a phenomenon where we write a for loop inside another for loop.

Syntax:

```
for var1 in collection:
    for var2 in collections:
        ...
            for varn in collection
                SB
```

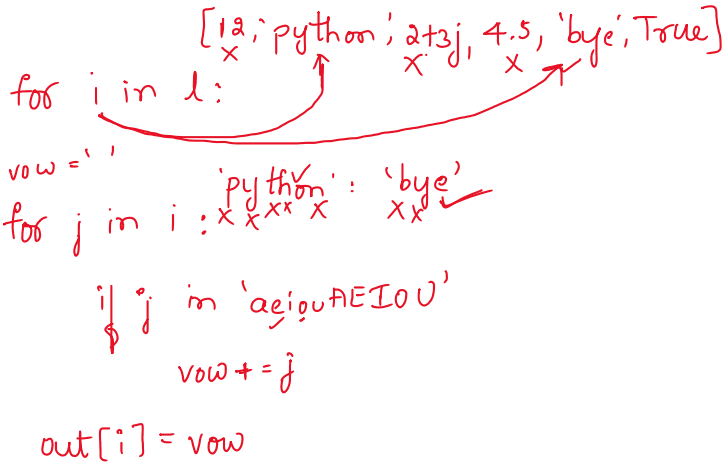
Programs

```
for i in range(1,5):
   for j in range(1,4):
      print(i,j)
```



Reason: First the variable will comes to the outer for loop and takes a value From the collection and enters to inner for loop . Until all the values of Inner for loop ends it will not go back to the outer for loop.

```
# Guess th following output.
'''
Input :  [12,'python',2+3j,4.5,'bye',True]
Output : {'python':'o','bye':'e'}    '''

'''
l = eval(input('Enter the list: '))
out = {}
for i in l:
   if type(i) == str:
      vow=''
      for j in i:
         if j in 'aeiouAEIOU':
            vow += j
      out[i] = vow
print(out)'''
```



```
# Guess the following output.
'''
Input :  [12,'python',2+3j,4.5,'bye',True]
Output : {'python':'pythn','bye':'by'}   '''

'''
l = eval(input('Enter the list: '))
out = {}
for i in l:
   if type(i) == str:
      cons=''
      for j in i:
         if j not in 'aeiouAEIOU':
            cons += j
      out[i] = cons
print(out)'''


# Guess the following output.
'''
Input :  [12,'python',2+3j,4.5,'bye',True]
Output : {'python':'PYTHON','bye':'BYE'}   '''
'''
l = eval(input('Enter the list: '))
out = {}
for i in l:
   if type(i) == str:
      upper=''
      for j in i:
         if 'a'<=j<='z':
            upper += chr(ord(j)-32)
         else:
            upper += j
      out[i] = upper
print(out) '''
```

Day-26

Strong Number: If the number is equal to the sum of the factorial of individual digits, the we call it as Strong number.

145 = 1! + 4! + 5!

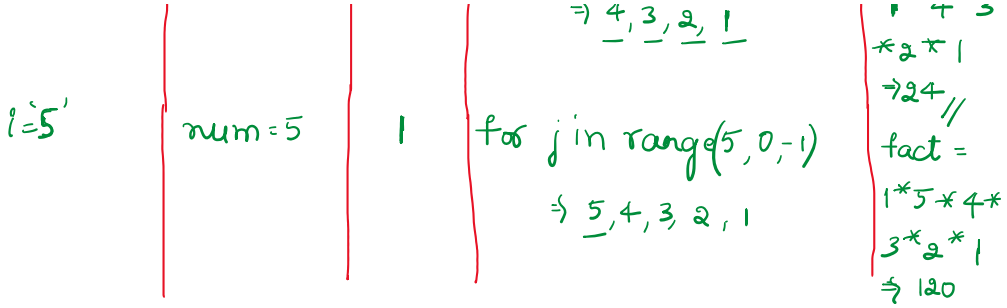
```
# WAP to check whether the given number is strong number or not
'''
n =  int(input('Enter the number: '))
sum_fact = 0
a = str(n)
for i in a:
   num = int(i)
   fact = 1
   for j in range(num,0,-1):
      fact *= j
   sum_fact += fact
if sum_fact == n:
   print('Strong Number')
else:
   print('Not strong number') '''
```

n ⇒ 145

a = str(n) = '145'

for i in a :	num = int(i)	fact = 1	for i in range(num, 0, -1)	fact* = ∫
i = '1'	num = 1	1	for i in range(1, 0, -1) ⇒ (1)	fact = 1*1 = 1
i = '4'	num = 4	1	for i in range(4, 0, -1) ⇒ 4, 3, 2, 1	fact = 1*4*3*2*1 ⇒ 24..

print('Not strong number') '''

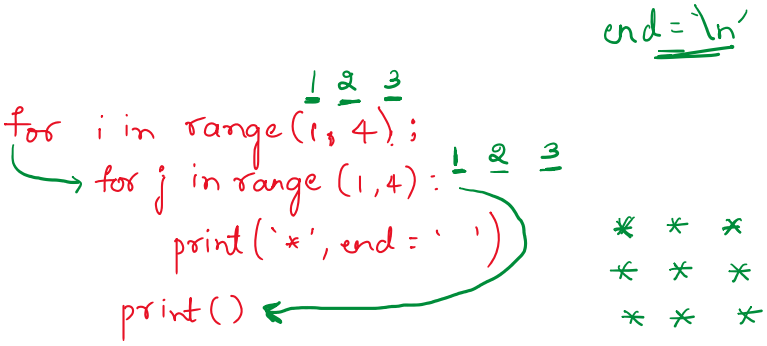


Patterns:

--- Using Nested for loop we print patterns.

Programs.

```
# Patterns
'''
* * *                    '''

'''
for i in range(1,4):
   print('*',end=' ') '''

'''
* * *
* * *
* * *                          '''

for i in range(1,4):
   for j in range(1,4):
      print('*', end=' ')
   print()
```

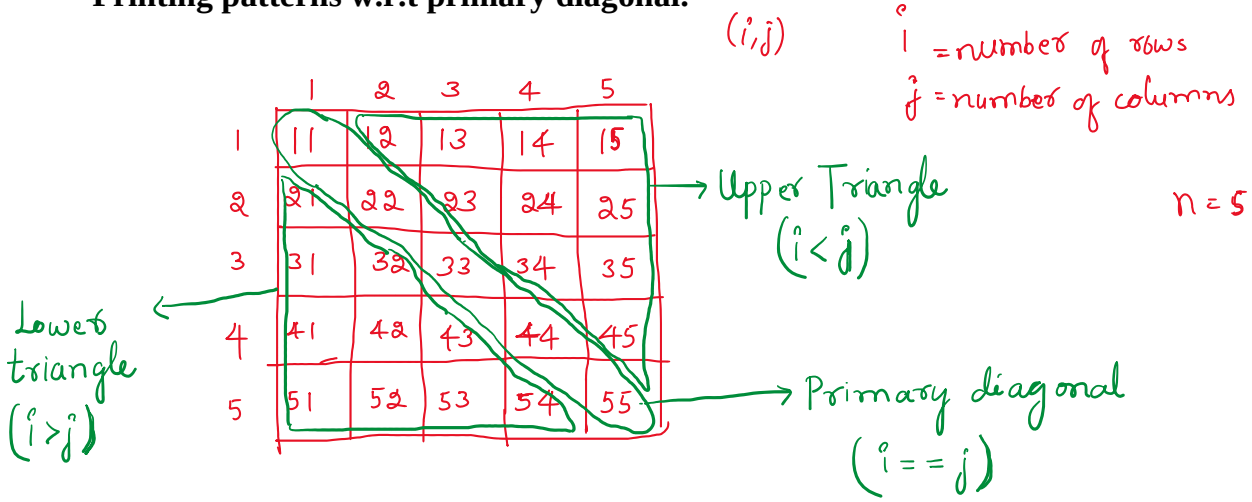


Day-28

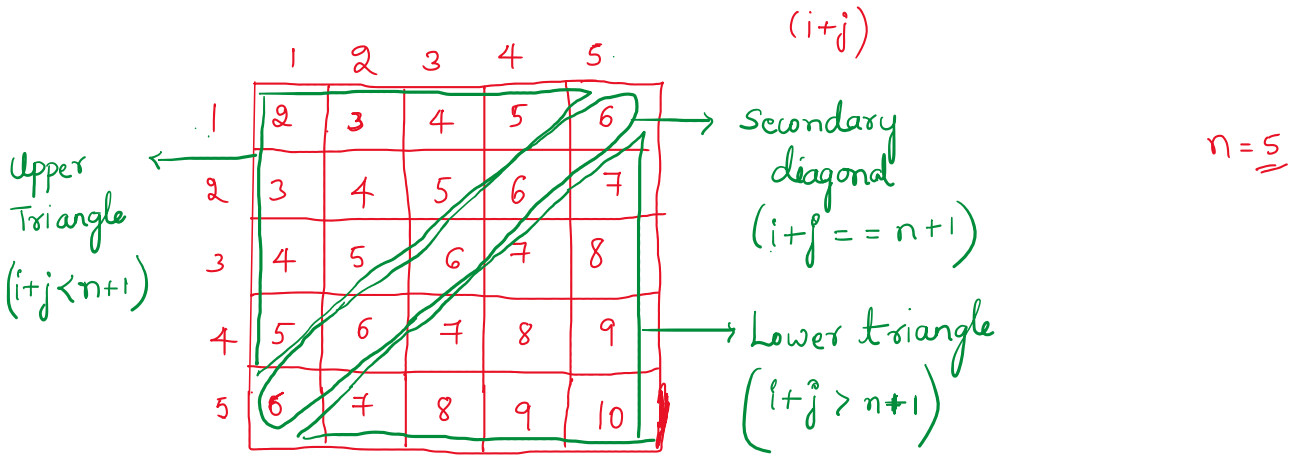
```
'''
* * *
* * *
* * *
* * *
* * *
for i in range(1,6):
   for j in range(1,4):
      print('*',end = ' ')
   print()

'''

'''
* * * * *
* * * * *
```

```
for i in range(1,3):
    for j in range(1,6):
        print('*',end = ' ')
    print()

'''

'''
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

n = int(input('Enter the number: '))
for i in range(1,n+1):
    for j in range(1,n+1):
        print('*', end = ' ')
    print()
'''
```

- **Printing patterns w.r.t primary diagonal.**



- **Printing patterns w.r.t secondary diagonal.**



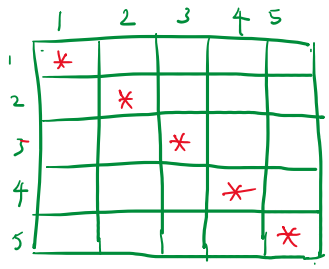
Programs:

```
'''
*
  *
    *
      *
        *
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
    for j in range(1,n+1):
        if i == j:
            print('*', end = ' ')
        else:
            print(' ',end = ' ')
    print()

'''
@
* @
* *  @
* * *  @
* * * *  @

n = int(input('Enter the number: '))
for i in range(1,n+1):
    for j in range(1,n+1):
        if i == j:
            print('@', end = ' ')
        elif i > j:
            print('*', end = ' ')
        else:
            print(' ',end = ' ')
    print() '''

'''
# # # # $
# # # $ &
# # $ &&
# $ &&&
$ &&&&

n = int(input('Enter the number: '))
for i in range(1,n+1):
    for j in range(1,n+1):
        if i+j == n+1:
            print('$', end = ' ')
        elif i+j > n+1:
            print('&', end = ' ')
        elif i+j < n+1:
            print('#', end = ' ')
    print()  '''

'''
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

n=5

1,2,3,4,5

```
for i in range(1,n+1):
    for j in range(1,n+1):
        if i==j:
            print('*', end=' ')
        else:
            print(' ', end=' ')
    print()
```



```
n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      if(i==j):
         print('1',end=' ')
      else:
         print('0', end=' ')
   print() '''

'''
* * * * *
*       *
*       *
*       *
* * * * *

n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      if i == 1 or i == n or j == 1 or j == n:
         print('*', end = ' ')
      else:
         print(' ', end = ' ')
   print() '''

'''
*     *
 * *
  *
 * *
*     *

n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      if i == j or i+j == n+1:
         print('*', end =' ')
      else:
         print(' ', end = ' ')
   print()
'''

'''
    *
    *
* * * * *
    *
    *

n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      if i == n//2+1 or j == n//2+1:
         print('*', end =' ')
      else:
         print(' ', end = ' ')
   print()
'''
```

Day-29

```
'''
* * * * * * * * *
* *   *   * *
*  *  *  *  *
*   * * *   *
* * * * * * * * *
*   * * *   *
*  *  *  *  *
* *   *   * *
* * * * * * * * *
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      if i == 1 or i == n or j == 1 or j == n or j == n//2+1 or \
      i == n//2+1 or i == j or i+j == n+1 :
         print('*', end = ' ')


'''
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      print(j,end = ' ')
   print()


'''
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
   for j in range(1,n+1):
      print(i,end = ' ')
   print()

'''
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
```

```
    for j in range(1,n+1):
        if i >= j :
            print(j,end = ' ')
        else:
            print(' ',end = ' ')

    print()

'''
23
23 24
23 24 25
23 24 25 26
23 24 25 26 27
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
    k = 23
    for j in range(1,n+1):
        if i >= j :
            print(k,end = ' ')
            k += 1
        else:
            print(' ',end = ' ')
    print()


'''
      5
     5 4
    5 4 3
  5 4 3 2
5 4 3 2 1
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
    k = 5
    for j in range(1,n+1):
        if  i+j == n+1 or i+j > n+1 :
            print(k,end = ' ')
            k -=1
        else:
            print(' ',end = ' ')
    print()

'''
A B C D E
 A B C D
  A B C
   A B
    A
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
    k = 'A'
    for j in range(1,n+1):
        if i <= j:
```

```
            print(k,end = ' ')
            k = chr(ord(k)+1)
        else:
            print(' ',end = ' ')
    print()

'''
L M N O P
 M N O P
  N O P
   O P
    P
'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
    k = ord('L') + i -1
    for j in range(1,n+1):
        if i <= j:
            print(chr(k),end = ' ')
            k += 1
        else:
            print(' ',end = ' ')
    print()

'''
 * * *
*     *
* * * * *
*     *
*     *

'''

n = int(input('Enter the number: '))
for i in range(1,n+1):
    for j in range(1,n+1):
        if (i == 1 and 1<j<n) or (j == 1 and i>1) or (j == n and i>1 ) or i==n//2+1:
            print('*',end = ' ')
        else:
            print(' ',end = ' ')
    print()


'''
* * * *
*     *
* * * *
*     *
* * * *

'''

n = 5
for i in range(1,n+1):
    for j in range(1,n+1):
        if j==1 or (i == 1 and j<n) or (i == n//2+1 and j<n ) or (i == n and j<n) or (j==n and i%2==0):
            print('*',end = ' ')
        else:
            print(' ',end = ' ')
```

```
print()
```

Day - 30

Intermediate Terminations in Looping.

--- The process of terminating the execution of loop in between and make the controller come out of the loop is called as **Intermediate Terminations in Looping.**

Types:

- **Break**
- **Continue**
- **Pass**

1) Break:

--- It can be used only for looping statements , whenever the controller sees the keyword break it will immediately stop the execution and come out of the loop.

Example:

```
# break

#Example
'''
num = int(input('Enter the number: '))
for i in range(1,11):
   print(i)
   if i ==num:
      break '''

# WAP to demonstrate guess the number game.
'''
n = 123
while True:
   num = int(input('Guess the number! '))
   if num == n:
      print('Congrats! you guessed it')
      break
   elif num > n:
      print('Entered number is greater')
   else:
      print('Entered number is lesser')'''

# WAP to check the given string is having only lowercase or not.
'''
s = input('Enter the string: ')
for i  in s:
   if not('a'<=i<='z'):
      print('It has other characters than lowercase')
      break
else:
   print('It has only lowercase character')'''
```

2) Continue:

--- Whenever we want to skip the current interation or particular iteration we use

continue keyword.

Example:

```
# Continue
'''
for i in range(1,11):
    if i==3 or i == 5 :
        continue
    print(i) '''

# WAP to extract all the given integers from list.
'''
l = eval(input('Enter the list: '))
out = []
for i in l:
    if type(i)!=int:
        continue
    else:
        out.append(i)
print(out) '''

# WAP to extract the special character from the string.
'''
s = input('Enter the string: ')
out = ''
for i in s:
    if 'a'<=i<='z' or 'A'<=i<='Z' or '0'<=i<='9':
        continue
    out += i
print(out) '''

# WAP to print 1 to 10 by skipping 3 and 8.
'''
i = 1
while i<=10:
    if i==3 or i==8:
        i+=1
        continue
    print(i)
    i+=1  '''

'''
for i in range(1,11):
    if i==3 or i==8:
        continue
    print(i) '''
```

3) Pass:

--- It is a keyword used to make any empty block as valid block.

Example:

```
# Pass
'''
a = 10
b = 20
if a>b:
    pass
```

```
print('hello everyone') '''
```

Functions:

--- It is a name given to a memory block where the instructions are stored and performing some specific task. By creating a function for one time we can use it for n time.

Why we need functions?

- We can reduce the number of instructions
- We can increase the efficiency of the code.
- We can avoid code repetition.
- Reuse the code for n number of times.

Types:

- Inbuilt functions
- User defined functions.

Inbuilt Functions.

--- They are the predefined set of instructions to perform specific task.

1) **Utility functions:** It is used for all datatypes.

   Example: id(), type(), bool(), eval(), print()

2) **Inbuilt function on String:**

   Example: upper(), lower(), swapcase(), replace(), capitalize(),title(),index(), count(), split(), join()

```
s = 'Python@123'
s.upper()
'PYTHON@123'
s = 'PYTHon@123'
s.lower()
'python@123'
s = 'PYTHon@123'
s.swapcase()
'pythON@123'
s = 'hello'
s.replace('l','b')
'hebbo'
s = 'pyTHon@123'
s.capitalize()
'Python@123'
s = 'hello my friends'
s.title()
'Hello My Friends'
s = 'hello my friends'
s.index('l')
2
s = 'hello my friends'
s.count('e')
2
s = 'hello my friends'
s.split()
['hello', 'my', 'friends']
```

```
s = 'hello my friends'
s.split('e')
['h', 'llo my fri', 'nds']
s = 'hello my friends'
s.join('1')
'1'
s
'hello my friends'
s = 'hello my friends'
s.join('1')
'1'
s
'hello my friends'
'8'.join('sakshi')
's8a8k8s8h8i'
' '.join('sakshi')
's a k s h i'
''.join('hello my friends')
'hello my friends'
s = 'hello my friends'
s.split()
['hello', 'my', 'friends']
a= s.split()
a
['hello', 'my', 'friends']
' '.join(a)
'hello my friends'
s = 'hellomyfriends'
s.split()
['hellomyfriends']
```

Day-31

3) Inbuilt function on List:

Example: append(), insert(), pop(), remove(), sort(), reverse(), count(), index()

```
l = [12,57,45,2.4,True]
l.sort()
l
[True, 2.4, 12, 45, 57]
l.sort(reverse = True)
l
[57, 45, 12, 2.4, True]
l = [12,57,45,2.4,True]
l.reverse()
l
[True, 2.4, 45, 57, 12]
l = [12,57,45,12,57,2.4,True]
l.count(57)
2
l.index(45)
2
```

4) Inbuilt function on Tuple:

Example: count(), index()

```
t = (12,57,45,12,57,2.4,True)
t.count(57)
2
t.index(45)
2
```

5) Inbuilt function on Set:

Example: add(), pop(), remove(), union(), intersection(), difference(), clear()

```
s = {12, 2.3,3.4, 6.8}
t = {12, 2.3, 97}
s.union(t)
{97, 2.3, 3.4, 6.8, 12}
s.intersection(t)
{2.3, 12}
s.difference(t)
{3.4, 6.8}
s.clear()
s
set()
```

6) Inbuilt function on Dictionary:

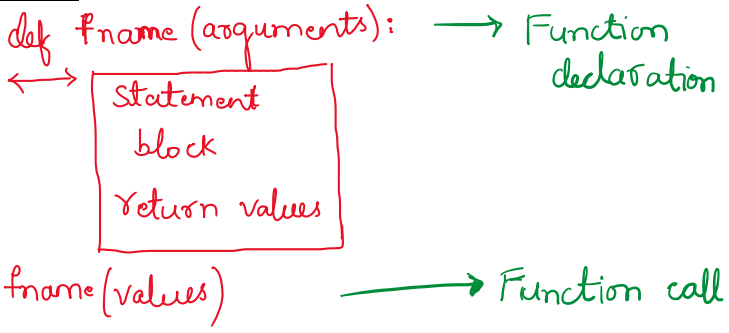
Example: keys(), values(), items(), pop(), get(), popitem(), update(), clear()

```
d = {'a':10, 'b':20 ,'c':30, 'd':40}
d.keys()
dict_keys(['a', 'b', 'c', 'd'])
d.values()
dict_values([10, 20, 30, 40])
d.items()
dict_items([('a', 10), ('b', 20), ('c', 30), ('d', 40)])
d.pop('d')
40
d
{'a': 10, 'b': 20, 'c': 30}
d.get('c')
30
d.popitem()
('c', 30)
d
{'a': 10, 'b': 20}
d.update()
d
{'a': 10, 'b': 20}

d.update(n)
d
{'a': 10, 'b': 20, 'm': 'monkey', 'o': 'oops'}
d.clear()
d
{}
```

User Defined Functions.

--- They are the functions which are defined by the user based on their requirements .

Syntax:



Note:

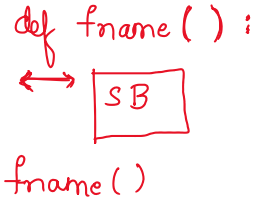
- **def** --- It is a keyword used to define the function.
- As soon as we create function it is not executed directly, in order to execute we have to call that function.
- Passing the arguments or return value are not manditory.
- Return keyword is used to stop the execution.

Types:

- Function without argument without return value.
- Function with argument without return value.
- Function without argument with return value.
- Function with argument with return value.

1) Function without argument without return value.

Syntax:

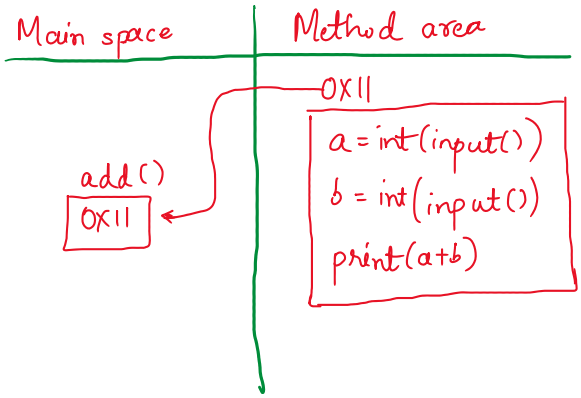


Example:

```
#Example program
'''
def add():
   a = int(input('Enter the number: '))
   b = int(input('Enter the number: '))
   print(a+b)
add() '''
```

Memory Allocation:

Memory Allocation:



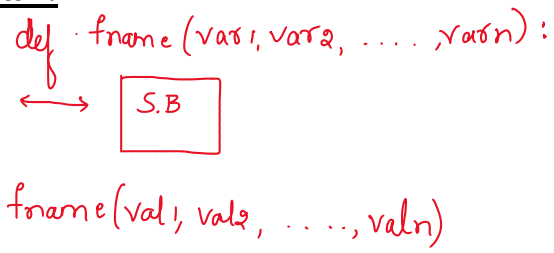
Programs:

```
# WAP to convert the string to uppercase.
'''
def convert():
    s = input('Enter the string: ')
    out = ''
    for i in s:
        if 'a'<=i<='z':
            out += chr(ord(i)-32)
        else:
            out += i
    print(out)
convert() '''

# program to count the number of occurence of  a given character in a given string.
'''
def count_char():
    s = input('Enter the string: ')
    ch = input('Enter the character: ')
    count = 0
    for i in s:
        if i == ch:
            count += 1
    print(count)
count_char()   '''
```

Note: **Arguments --- They are generally represented as value holders.**

2) Function with argument without return value.

Syntax:




```
# Example
'''
def add(a,b):
```

```
    print(a+b)
add(int(input('Enter the num: ')),int(input('Enter the num: ')))
'''

# WAP to convert the string to uppercase.
'''
def convert(s,out):
    for i in s:
        if 'a'<=i<='z':
            out += chr(ord(i)-32)
        else:
            out += i
    print(out)
convert(input('Enter the string: '),'')'''

# program to count the number of occurence of  a given character in a given string.
'''
def count_char(s,ch,count):
    for i in s:
        if i == ch:
            count += 1
    print(count)
count_char(input('Enter the string: '),input('Enter the character: '),0)
'''
```

3) Function without argument with return value.

Syntax:

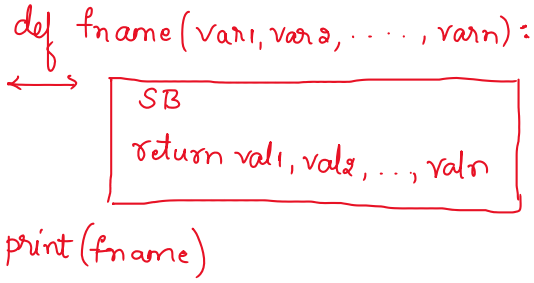
```
def fname( ):
    SB
    return val1, val2, ....., valn

var1, var2, var3 ...varn = fname( )
        (or)
var = fname ( )
        (or)
print (fname)
```

```
# Function without argument with return value.

#Example program
'''
def add():
    a = int(input('Enter the number: '))
    b = int(input('Enter the number: '))
    return (a+b)
print(add()) '''

# WAP to convert the string to uppercase.
'''
def convert():
    s = input('Enter the string: ')
```

```
    out = ''
    for i in s:
        if 'a'<=i<='z':
            out += chr(ord(i)-32)
        else:
            out += i
    return out
print(convert()) '''


# program to count the number of occurence of  a given character in a given string.
'''
def count_char():
    s = input('Enter the string: ')
    ch = input('Enter the character: ')
    count = 0
    for i in s:
        if i == ch:
            count += 1
    return count
print(count_char())  '''
```

4) Function with argument with return value.

Syntax:



Programs:

```
# Example
'''
def add(a,b):
    return a+b
print(add(int(input('Enter the num: ')),int(input('Enter the num: '))) )'''

# WAP to convert the string to uppercase.
'''
def convert(s,out):
    for i in s:
        if 'a'<=i<='z':
            out += chr(ord(i)-32)
        else:
            out += i
    return out
print(convert(input('Enter the string: '),''))
'''


# program to count the number of occurence of  a given character in a given string.
'''
def count_char(s,ch,count):
```

```
    for i in s:
        if i == ch:
            count += 1
    return count
print(count_char(input('Enter the string: '),input('Enter the character: '),0))
'''
```

Day-32

Global and Local Variable:

1) Global Variable:

--- These are the variables which are created in main space and which can be accessed and modified in the main space but we can only access them inside the method area.

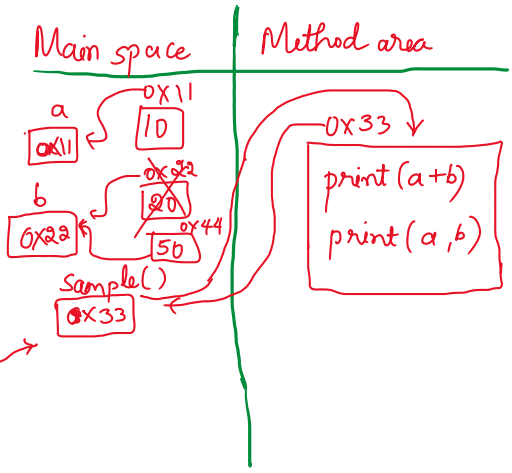
Example:

```
a = 10
b = 20
def sample():
    print(a+b)
    print(a,b)
print(a+b)
sample()
b = 50
print(a,b)
```

Output

```
30
30
10 20
10 50
```

Memory allocation:



Note:

--- We have to use global keyword to do modification for the global variables inside the function.

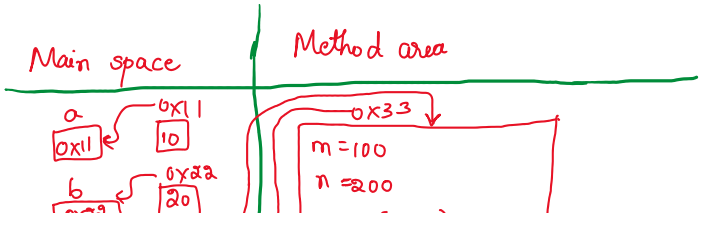
```
a = 10
b = 20
def sample():
    global a
    print(a+b)
    a = 40
    print(a+b)
print(a+b)
sample()
b = 50
print(a,b)
```

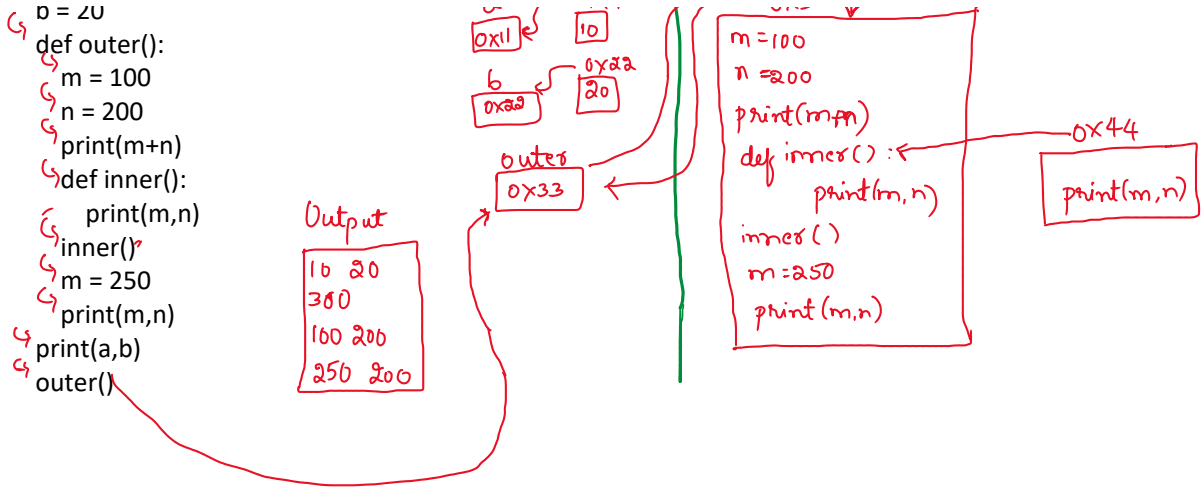
2) Local Variable:

--- These are the variables which are created in method area and which can be accessed and modified in the same function but we can only access them inside the nested function.

Example:

```
a = 10
b = 20
def outer():
    m = 100
```



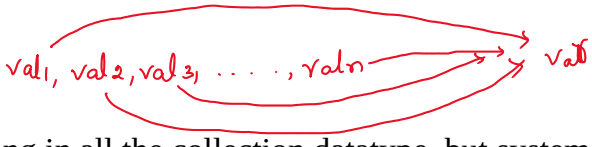


Note:
--- We have to use nonlocal keyword to do modification for the local variables inside the nested function.

```
'''
a = 10
b = 20
def outer():
   m = 100
   n = 200
   print(m+n)
   def inner():
      nonlocal n
      print(m,n)
      n = 300
      print(m,n)
   inner()
   m = 250
   print(m,n)
print(a,b)
outer()
'''
```

Packing and Unpacking:

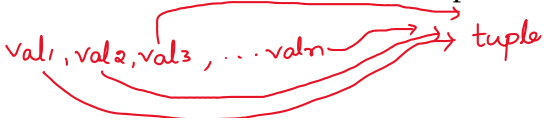
Packing:
--- It is a phenomenon of grouping the individual values in the form of collections to provide security.



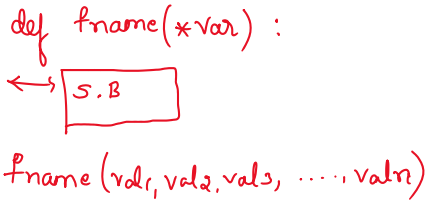
Note:
-- We can do packing in all the collection datatype, but system will do packing in only tuple datatype.

Types:
- Single / Tuple packing
- Double / Dictionary packing.

1) Single / Tuple packing:
--- Packing individual values in the form of tuple.



Syntax:

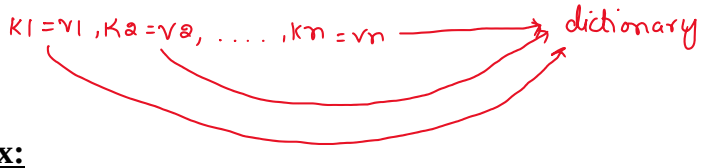


Program:

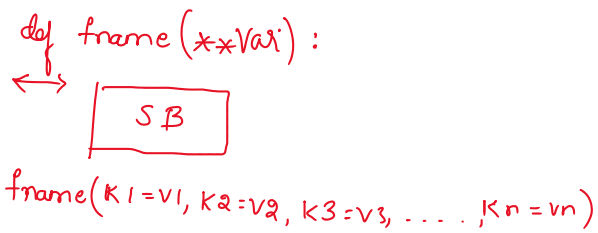
```
# Tuple packing
'''
def pack(*a):
   print(type(a))
   print(a)
pack(10,20,30,2.3,4+2j)
'''
'''
def pack(*s):
   print(type(s))
   print(s)
pack(2.3,3,6,5)
'''
```

1) Double / Dictionary packing:

--- Packing individual values in the form of dictionary.



Syntax:



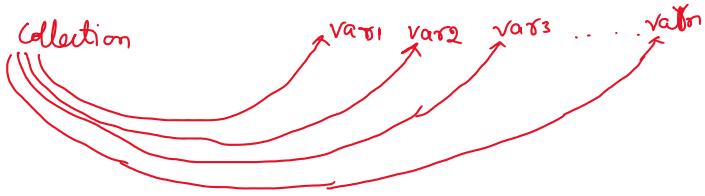

```
# Combination of single and double packing.
'''
def pack(*t, **d):
   print(type(t))
   print(t)
   print(type(d))
   print(d)
pack(1,'two',3.0,a=10,b=20,c=30)'''
```

Note:

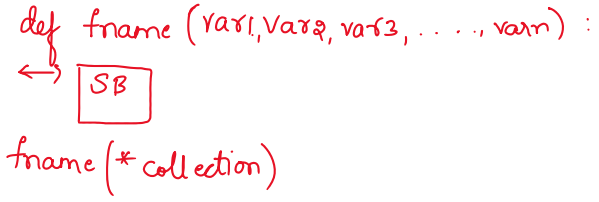
- Keys should follow the rules of identifier

- Key-value pair should get seperated by =
- Keys should be in string.
- Keys shouldn't include the quotes , but values must be enclosed with quotes.

Unpacking:
--- The phenomenon of dividing the collection into individual values and storing every single value in a completely unique variable.



Syntax:



Programs:

```
# Unpacking.
'''
def unpack(a,b,c,d):
   print(a)
   print(b)
   print(c)
   print(d)
unpack(*'1234')'''

'''
def unpack(a,b,c,d):
   print(a)
   print(b)
   print(c)
   print(d)
unpack(*[10,20,30,40]) '''

'''
def unpack(a,b,c,d):
   print(a)
   print(b)
```

```
    print(c)
    print(d)
unpack(*(10,20,30,40)) '''

'''
def unpack(a,b,c,d):
    print(a)
    print(b)
    print(c)
    print(d)
unpack(*{10,20,30,40}) '''

'''
def unpack(a,b,c,d):
    print(a)
    print(b)
    print(c)
    print(d)
unpack(*{'1':10,'2':20,'3':30,'4':40}) '''

'''
def unpack(a,b,c,d):
    print(a)
    print(b)
    print(c)
    print(d)
unpack(*{'1':10,'2':20,'3':30,'4':40}.values())'''

'''
def unpack(a,b,c,d):
    print(a)
    print(b)
    print(c)
    print(d)
unpack(*{'1':10,'2':20,'3':30,'4':40}.items()) '''

'''
def unpack(a,b,c,d):
    print(a)
    print(b)
    print(c)
    print(d)
unpack(*range(1,5)) '''
```

Day-33

Types of Arguments:

1) Positional argument:

--- These are present in function declaration .

Rules:

- **It is compulsory to follow the order**
- **It is compulsory to pass the value.**

Example:

```
def count_char(s,ch,count):
   for i in s:
      if i == ch:
         count += 1
   print(count)
count_char(input('Enter the string: '),input('Enter the character: '),0)
'''
```

2) Default argument:

--- These are present in function declaration. If user is not passing the value it will consider the default value or else it will consider the entered value.

Rule:

- It is not compulsory to pass the value.
- If we pass the value, it will consider that value or else it will consider the default value.

Example:

```
def add(a=0,b=0):
   print(a+b)
add()
```

3) Keyword argument:

--- These are present in function call. They are present in the form of key-value pair.

Example:

```
def pack(**d):
   print(type(d))
   print(d)
pack(a=10,b=20,c=30)
```

4) Variable Length Argument:

--- They are function declaration which are capable of storing values from 1 to n.

Example:

```
def pack(*a):
   print(type(a))
   print(a)
pack(10,20,30,2.3,4+2j)
```

Recursion:

--- The phenomenon of calling the function by itself is called as Recursion until the termination condition becomes True.

Syntax:

1) Without return value

```
def fname(args):
    if termination_cond:
```

2) With return value

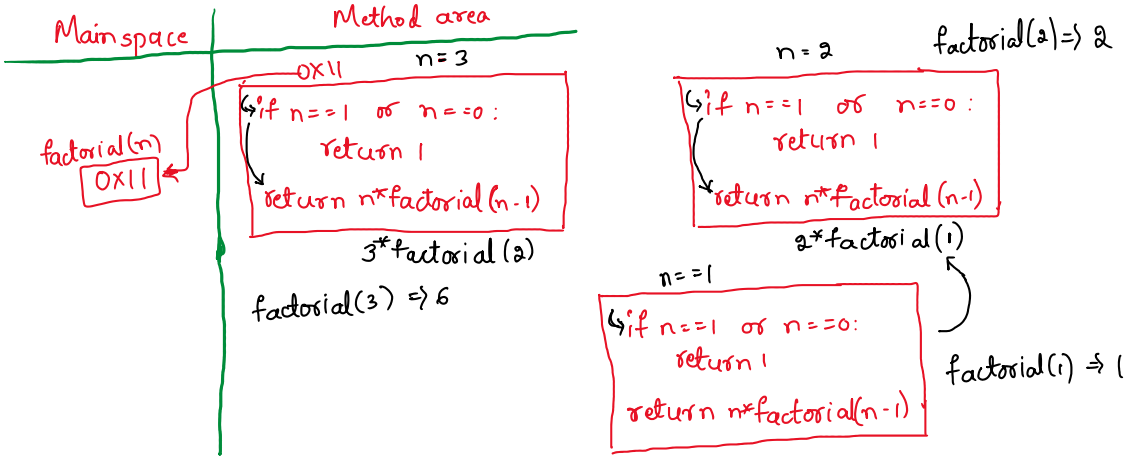
```
def fname(args):
    if termination_cond:
```

```
def fname(args):
    if termination-cond:
        return
    fname(args)
fname(val)
```

```
def fname(args):
    if termination_cond:
        return values
    return fname(args)
print(fname(values))
```

Programs:

```
# WAP to find the factorial of a given number:
'''
def factorial(n):
    if n == 1 or n == 0:
        return 1
    return n*factorial(n-1)
print(factorial(3)) '''
```




```
# WAP to find the sum of n natural numbers.
'''
def sum(n):
    if n == 1:
        return 1
    return n+sum(n-1)
print(sum(int(input('Enter the number: ')))) '''
```

Steps to convert looping program into recursion:

1) Initialisation of all the looping variable should be done in the function declaration. Section. ✓
2) The termination condition should be written exactly opposite to looping condition in the form of if statement. ✓
3) Return the total result inside the termination condition ✓
4) Logic of the program should be written as it is ✓
5) Updation of looping variable should be done in recursive call.

Programs:

< ⟹ >=
> ⟹ <=

Recursion:

```
def lower(s,out='',i=0):
    if i>=len(s):
        return out
    if 'a'<=s[i]<='z':
        out+=s[i]
    return lower(s,out,i+1)
print(lower(input('enter the string:')))
```

while loop

```
s=input('enter the string:')
out=''
i=0
while i<len(s):
    if 'a'<=s[i]<='z':
        out+=s[i]
    i+=1
print(out)
```

```
# Extract all the lowercase from the string.
'''
def lower(s,out='',i=0):
    if i>=len(s):
        return out
    if 'a'<=s[i]<='z':
        out += s[i]
    return lower(s,out,i+1)
print(lower(input('Enter the string:'))) '''
```

Day-34

```
# Get the following output
# input = ['hai',56,7+8j,45,8.7,'data']
# output = ['iahhai',56,7+8j,45,8.7,'ataddata']
```

Recursion

```
def result(l,out=[],i=0):
    if i>=len(l):
        return out
    if type(l[i])==str:
        out.append(l[i][::-1]+l[i])
    else:
        out.append(l[i])
    return result(l,out,i+1)
print(result(eval(input())))
```

while loop

```
l=eval(input('enter the list:'))
out=[]
i=0
while i<len(l):
    if type(l[i])==str:
        out.append(l[i][::-1]+l[i])
    else:
        out.append(l[i])
    i+=1
print(out)
```

```
# Get the following output
# input = ['hai',56,7+8j,45,8.7,'data']
# output = ['iahhai',56,7+8j,45,8.7,'ataddata']
'''
def result(l,out=[],i=0):
```

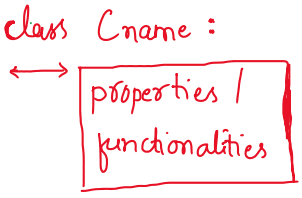
```
    if i>=len(l):
        return out
    if type(l[i]) == str:
        out.append(l[i][::-1]+l[i])
    else:
        out.append(l[i])
    return result(l,out,i+1)
print(result(eval(input('Enter the list: ')))) '''

# Assignment
# Select any 3 while loop programs and convert to recursion.
```

OOPS:
--- The concepts which deals with the class and objects .

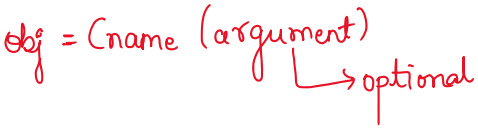
Class: It is a container which is used to store the properties and functionalities of a real time entity.
Or
It is a container which will allow the user to store the data and it allows the user to utilize the data.

Object: It is an instance of class.

Class Creation:



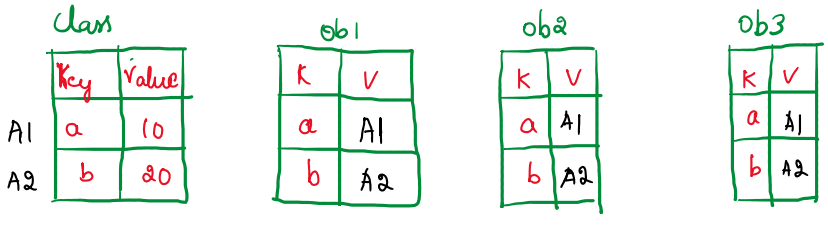
Object Creation:



Example:

```
class Demo:
    a = 10
    b = 20
ob1 = Demo()
ob2 = Demo()
ob3 = Demo()
```

Memory Allocation:



Whenever we want to access value from class or object,
For class ---- Cname.var
For object ---- obj.var

```
class Demo:
   a = 10
   b = 20
ob1 = Demo()
ob2 = Demo()
ob3 = Demo()
print(Demo.a,Demo.b)
print(ob1.a,ob1.b)
print(ob2.a,ob2.b)
print(ob3.a,ob3.b)
```

Example:

```
# Create a class Bank and 2 objects
'''
class Bank:
   bname = 'SBI'
   loc = 'Bangalore'
c1 = Bank()
c2 = Bank()

print(Bank.bname,Bank.loc)
print(c1.bname,c1.loc)
print(c2.bname, c2.loc)

print('*'*50)

Bank.loc = 'Mumbai'
print(Bank.bname,Bank.loc)
print(c1.bname,c1.loc)
print(c2.bname, c2.loc)

print('*'*50)

c1.loc = 'kochi'
print(Bank.bname,Bank.loc)
print(c1.bname,c1.loc)
print(c2.bname, c2.loc)

print('*'*50)

c2.loc = 'Vishakapatnam'
print(Bank.bname,Bank.loc)
print(c1.bname,c1.loc)
print(c2.bname, c2.loc) '''
```

Note:

- Modification done w.r.t class will affect to all the objects and that class too.
  Reason: Objects are an instance of class

- Modification done w.r.t object will affect only that particular object and rest of the objects and class will remain as it is.
  Reason: class and other objects are not depending on the objects.

Day-35

States:
--- The properties or functionalities of the class are defined as States.

Types:

- Generic / static / class
- Specific / object

1) **Generic / static / class** :
   --- The members of the class which are common for each and every object we create are called Generic states.

   Example: Sname, Loc, School timings, Principal, email_id, contact, uniform

Program:

```
# Generic State
'''
class School:
   Sname = 'St.Joseph'
   loc  = 'Jayanagar'
   website = 'www.sj.com'
   principal = 'Rancho'
st1 = School()
st2 = School()
print(School.Sname, School.loc,School.website,School.principal)
print(st1.Sname, st1.loc,st1.website,st1.principal)
print(st2.Sname, st2.loc,st2.website,st2.principal) '''
```

1) **Specific / object** :
   --- The members of the object which are created outside the class after the object creation.

   Program:

```
# Specific State

class School:
   Sname = 'St.Joseph'
   loc  = 'Jayanagar'
   website = 'www.sj.com'
   principal = 'Rancho'
st1 = School()
st2 = School()
print(School.Sname, School.loc, School.website, School.principal)
st1.name = 'A'
st1.id = '21'
st1.addr = 'Vijayanagar'
print(st1.name, st1.id, st1.addr)
st2.name = 'B'
st2.id = '22'
st2.addr = 'Whitefield'
print(st2.name, st2.id, st2.addr)
```

What is the difference between Functions and methods?

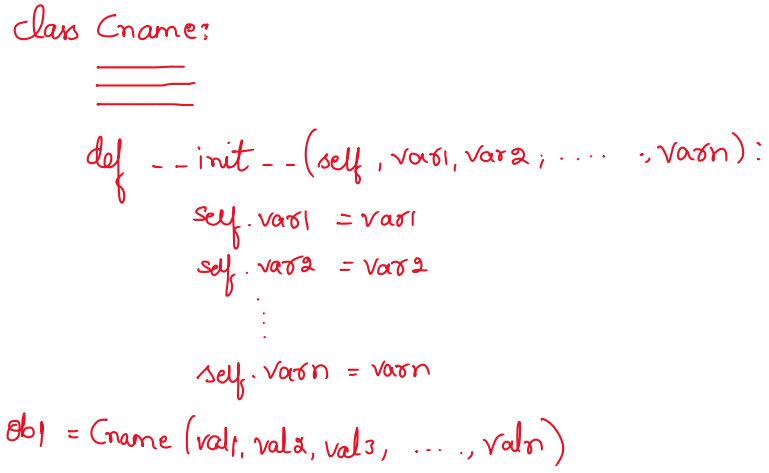
--- Functions defined outside the class are called as Functions.
Functions defined inside the class are called as Methods.

Constructor / __init__ / Initialisation :

- It is used to initialise the members of the object.
- No need of calling this __init__ method by default it will be called during object creation.
- We have to pass one argument for the __init__ method --- ' self '
  Self --- It is used to store the address of the object.

- We can pass the value for all the arguments in the object creation only if there is __init__ method.

Syntax:




```
# Constructor / __init__ / Initialization.
'''
class School:
   Sname = 'St.Joseph'
   loc  = 'Jayanagar'
   website = 'www.sj.com'
   principal = 'Rancho'
   def __init__(self,name,sid,addr):
      self.name = name
      self.sid = sid
      self.addr = addr
st1 = School('A',21,'Mysore')
st2 = School('B',22,'bangalore')

print(st1.name, st1.sid, st1.addr)
print(st2.name, st2.sid, st2.addr)'''
```

Assignment : To create a class Company with 3 class members , 1 object and 4 object members

Methods:
--- Functions defined inside the class .

Types:

- Object method
- Class method
- Static method

1) Object method :
--- They are used to access and modify the members of the object.

Example:
To access

```
# Object method
'''
class Company:
   Cname = 'TCS'
   loc  = 'Jayanagar'
   website = 'www.tcs.com'
   def __init__(self,ename,eid,esal):
      self.ename = ename
      self.eid = eid
      self.esal = esal
   def display(self):
      print(self.ename,self.eid,self.esal)

emp1 = Company('A',33,60000)
emp1.display() '''
```

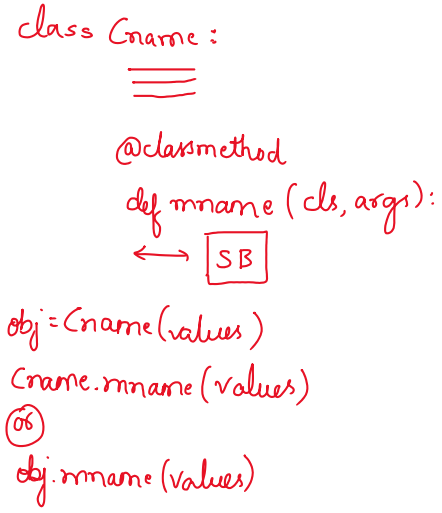
To modify:

```
# Object method

class Company:
   Cname = 'TCS'
   loc  = 'Jayanagar'
   website = 'www.tcs.com'
   def __init__(self,ename,eid,esal):
      self.ename = ename
      self.eid = eid
      self.esal = esal
   def display(self):
      print(self.ename,self.eid,self.esal)
   def ch_esal(self,new):
      self.esal = new

emp1 = Company('A',33,60000)
emp1.display()
emp1.ch_esal(70000)
emp1.display()
```

Day-36

2) <u>Class method</u> :
--- They are used to access and modify the members of the class. For all the class method we need to pass ' cls ' as an argument which is used to store the address of the class. To create a class it is compulsory to use the decorator **@classmethod**

Syntax:

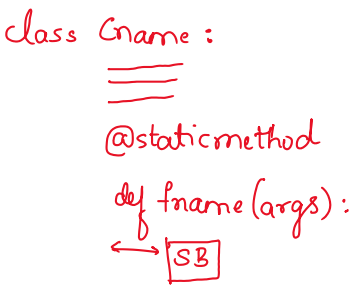


Example:

```
# class method
'''
class Company:
    Cname = 'TCS'
    loc  = 'Jayanagar'
    website = 'www.tcs.com'
    def __init__(self,ename,eid,esal):
        self.ename = ename
        self.eid = eid
        self.esal = esal
    def display(self):
        print(self.ename,self.eid,self.esal)
    def ch_esal(self,new):
        self.esal = new
    @classmethod
    def disp(cls):
        print(cls.Cname,cls.loc,cls.website)
    @classmethod
    def ch_loc(cls,new):
        cls.loc = new
emp1 = Company('Akash',21,37000)
Company.disp()
Company.ch_loc('Vijayanagar')
Company.disp()
'''
```

3) <u>Static method</u> :
--- It is neither related to class members nor object members but it will acts as supporting method for both class and object method.

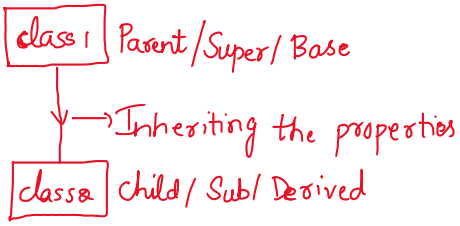
Syntax:



Program:

```
# Static method.
'''
class Bank:
   bname = 'SBI'
   loc = 'Pune'
   helpline = 98765
   def __init__(self,uname,uifsc,uacc,bal=500):
      self.uname = uname
      self.uifsc   = uifsc
      self.uacc   = uacc
      self.bal     = bal
   def display(self):
       print(self.uname,self.uifsc,self.uacc,self.bal)
   def change_acctype(self,new):
      self.uacc = new

   def deposit(self,amt):
      self.bal = self.bal+amt

   def withdraw(self,amt):
      if amt<self.bal:
         self.bal = self.bal - amt
      else:
         print('Insufficient balance')

   @classmethod
   def display1(cls):
      print(cls.bname,cls.loc,cls.helpline)
   @classmethod
   def ch_helpline(cls,new):
      cls.helpline = new
   @staticmethod
   def add(a,b):
      return a+b
   @staticmethod
   def sub(a,b):
      return a-b
user1 = Bank('Arun',4835348,'saving',)
user1.display()
user1.deposit(1000)
user1.display()
user1.withdraw(200)
user1.display()
user1.add(10,20)
user1.sub(10,20)   '''
```

Inheritance:
--- The phenomenon of deriving the properties of one class to another.



Why we need Inheritance?

- Reduce the number of instructions
- Increase the efficiency.

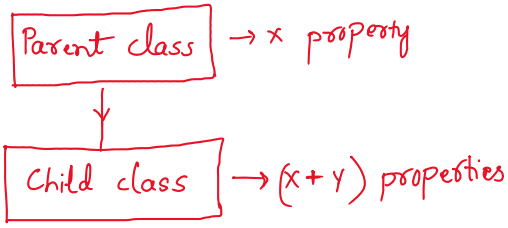
Parent class: The class from which we inherit the properties .

Child class: The class to which we inherit the properties.

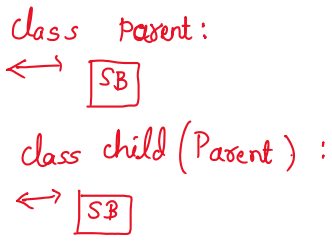
Types:

- Single level
- Multi-level
- Multiple
- Heirarchical
- Hybrid

1) Single level :
--- The phenomenon of inheriting the properties from one parent class to one child class.



Syntax:



Program:

```
#Example
'''
class A:
```

```
    a=10
    b=20
    @classmethod
    def disp(cls):
        print(cls.a,cls.b)
ob= A()
class B(A):
    c = 30
    d=40
    @classmethod
    def display(cls):
        print(cls.a,cls.b,cls.c,cls.d)
ob=B()
A.disp()
B.display() '''

# Program

class Bank:
    bname  = 'SBI'
    loc = 'Kerala'
    def __init__(self,name,phno,addr):
        self.name = name
        self.phno = phno
        self.addr = addr
    def display(self):
        print(self.name,self.phno,self.addr)
    @classmethod
    def disp(cls):
        print(cls.bname,cls.loc)
user1 = Bank('Ayush',95738465,'jaipur')
class Bank_updated(Bank):
    def __init__(self,name,phno,addr,pan,bal):
        self.name = name
        self.phno = phno
        self.addr = addr
        self.pan = pan
        self.bal = bal
    def display1(self):
        print(self.name,self.phno,self.addr,self.pan,self.bal)
user2 = Bank_updated('Ayush',95738465,'jaipur',34356,5000)
user2.display1()
```

Day-37

Constructor chaining:

--- we will be calling the constructor of parent class in the constructor of child class.

super().__init__(args)

Method chaining:

--- we will be calling the method of parent class in the method of child class.

super().method(args)

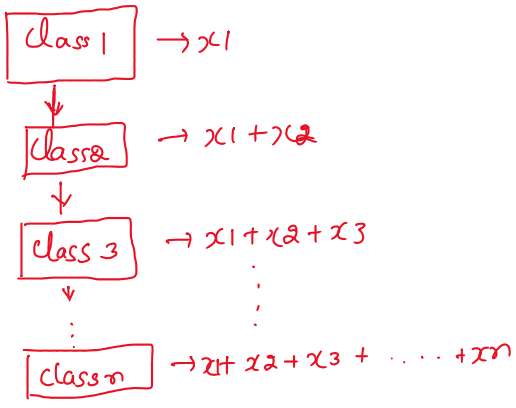
Example:

```
# Constructor chaining and method chaining
'''
```

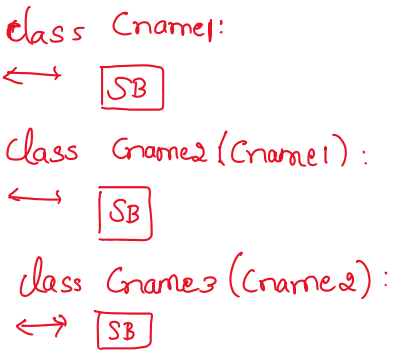
```
class Bank:
   bname  = 'SBI'
   loc = 'Kerala'
   def __init__(self,name,phno,addr):
      self.name = name
      self.phno = phno
      self.addr = addr
   def display(self):
      print(self.name,self.phno,self.addr)
   @classmethod
   def disp(cls):
      print(cls.bname,cls.loc)
user1 = Bank('Ayush',95738465,'jaipur')
class Bank_updated(Bank):
   def __init__(self,name,phno,addr,pan,bal):
      super().__init__(name,phno,addr)
      self.pan = pan
      self.bal = bal
   def display1(self):
      print(self.pan,self.bal,end = ' ')
      super().display()
user2 = Bank_updated('Ayush',95738465,'jaipur',34356,5000)
user2.display1()  '''
```

2) Multi-level :

--- Deriving the properties from parent class to child class takes place in multiple levels.



Syntax :

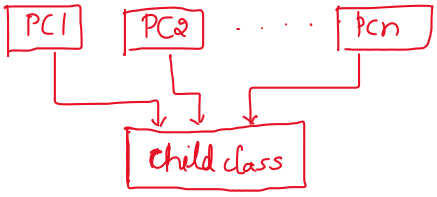



```
# Multi-level Inheritance
'''
class Bank:
   bname  = 'ICICI'
   loc = 'Patna'
   app = 'iMobile'
```

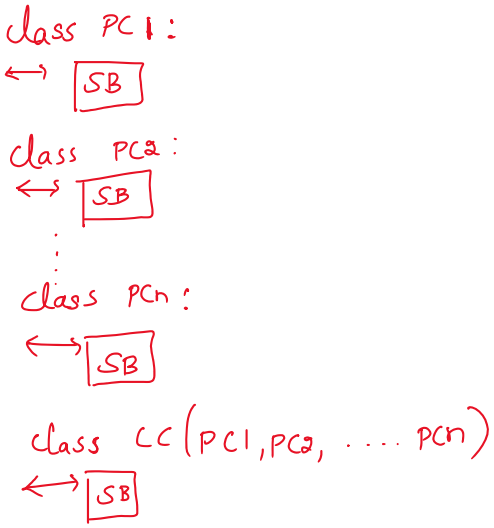
```
    @classmethod
    def display(cls):
        print(cls.bname,cls.loc,cls.app)
class Bank1(Bank):
    website = 'www.icici.com'
    irate = 40
    @classmethod
    def display1(cls):
        super().display()
        print(cls.website,cls.irate)
class Bank2(Bank1):
    manager = 'Scott'
    established = 1998
    @classmethod
    def display2(cls):
        super().display1()
        print(cls.manager,cls.established)
Bank2.display2() '''
```

3) Multiple Inheritance:

--- Inheriting properties from multiple parent class to a single child.



Syntax:



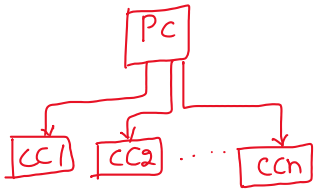
Programs:

```
# Multiple Inheritance
'''
class Bank:
    bname  = 'ICICI'
    loc = 'Patna'
    app = 'iMobile'
    @classmethod
```

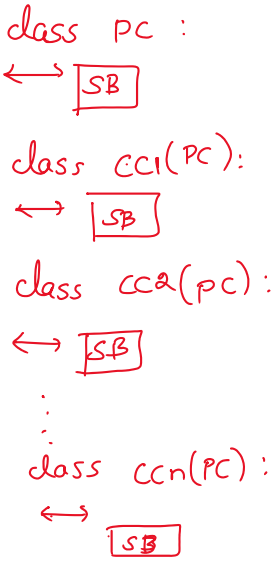
```
    def display(cls):
        print(cls.bname,cls.loc,cls.app)
class Bank1:
    website = 'www.icici.com'
    irate = 40
    @classmethod
    def display1(cls):
        print(cls.website,cls.irate)
class Bank2(Bank1,Bank):
    manager = 'Scott'
    established = 1998
    @classmethod
    def display2(cls):
        super().display()
        super().display1()
        print(cls.manager,cls.established)
Bank2.display2()    '''
```

4) Heirarchical Inheritance:

--- The phenomenon of inheriting the properties from single parent class to multiple child class.



Syntax:



Program:

```
# Heirarchical Inheritance

class Bank:
    bname  = 'ICICI'
    loc = 'Patna'
    app = 'iMobile'
    @classmethod
    def display(cls):
```

```
        print(cls.bname,cls.loc,cls.app)
class Bank1(Bank):
    website = 'www.icici.com'
    irate = 40
    @classmethod
    def display1(cls):
        super().display()
        print(cls.website,cls.irate)
class Bank2(Bank):
    manager = 'Scott'
    established = 1998
    @classmethod
    def display2(cls):
        super().display()
        print(cls.manager,cls.established)

Bank1.display1()
Bank2.display2()
```

5) Hybrid Inheritance:

--- The combination of more than one type of inheritance is called as Hybrid Inheritance.

Program:

```
# Hybrid Inheritance.

class Bank:
    bname  = 'ICICI'
    loc = 'Patna'
    app = 'iMobile'
    @classmethod
    def display(cls):
        print(cls.bname,cls.loc,cls.app)
class Bank1(Bank):
    website = 'www.icici.com'
    irate = 40
    @classmethod
    def display1(cls):
        super().display()
        print(cls.website,cls.irate)
class Bank2:
    manager = 'Scott'
    established = 1998
    @classmethod
    def display2(cls):
        print(cls.manager,cls.established)

class Bank3(Bank2,Bank1):
    helpline = 95863281
    @classmethod
    def display3(cls):
        super().display1()
        super().display2()
        print(cls.helpline)

Bank1.display1()
print('*'*50)
```

```
Bank3.display3()
```

ot

07 March 2025 07:43